50325-0512
(Seq. No. 3245 / CPOL 86661) *Patent*

UNITED STATES PATENT APPLICATION

FOR

STREAM CIPHER ENCRYPTION METHOD AND APPARATUS THAT CAN EFFICIENTLY SEEK TO ARBITRARY LOCATIONS IN A KEYSTREAM

INVENTORS:

DAVID A. MCGREW
SCOTT R. FLUHRER

PREPARED BY:

HICKMAN, PALERMO, TRUONG & BECKER
1600 WILLOW STREET
SAN JOSE, CA 95125
(408) 414-1080

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number: EL624356549US

Date of Deposit September 29, 2000

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

CASEY MOORE
(Typed or printed name of person mailing paper or fee)

Casey Moore
(Signature of person mailing paper or fee)

STREAM CIPHER ENCRYPTION METHOD AND APPARATUS THAT CAN EFFICIENTLY SEEK TO ARBITRARY LOCATIONS IN A KEYSTREAM

FIELD OF INVENTION

The present invention generally relates to cryptology. The invention relates more specifically to a stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream, including a method of generating an arbitrary segment of keystream for use by a stream cipher in encryption or decryption.

BACKGROUND OF THE INVENTION

Stream ciphers convert a plaintext to a ciphertext one bit at a time. In general, a stream cipher has a keystream generator that outputs a keystream consisting of a series of bits that, for perfect security, vary in value in an unpredictable manner. Each keystream bit is combined using a Boolean exclusive-OR operation (XOR) with an incoming bit of the plaintext, resulting in an output bit of the ciphertext. Thus, an additive stream cipher encrypts a plaintext by bitwise adding a pseudo-randomly generated keystream into the plaintext, modulo two.

For decryption, the ciphertext bits are XORed with an identical keystream to recover the plaintext bits. Accordingly, a stream cipher is ideally suited to encrypting a continuing stream of data, such as the data passing over a network connection between two computers or other network elements. Also, the security of a stream cipher resides in the randomness of the keystream, however, the keystream must be reproducible in identical form at decryption time. Therefore, design of the keystream generator is essential to security and practical operation.

FIG. 1A is a simplified block diagram of a stream cipher. A key *K* 401 is fed to keystream generator 402, which outputs keystream 410. Plaintext 412 is encrypted by an encryption function 416 based on keystream 410. As a result, ciphertext output 414 is produced.

The keystream generator of such a stream cipher can be described in terms of a state update function and an output function. For example, in FIG. 1A, keystream generator 402 has internal state information 404, a next state function 406 (state update function), and output function 408. The state update function maps the internal state of the keystream generator at one instant to its next value. The output function maps the internal state to a segment of keystream, and the keystream is defined as the concatenation of the values of the output function. Further background information on stream ciphers is provided in B. Schneier, "Applied Cryptography: Protocols, Algorithms and Source Code in C," 2nd ed. (New York: John Wiley & Sons, 1996).

Block ciphers such as the Data Encryption Standard (DES) are popularly used for encryption of computer communications. However, empirical evidence indicates that stream ciphers are faster than block ciphers at equivalent security levels. For example, in practical evaluation, the stream ciphers RC4 and SEAL have been determined to be significantly faster than any secure block cipher when implemented on general-purpose computer processors. Further, and RC4 and SEAL have survived years of scrutiny by cryptanalysts. SEAL is described in U.S. Pat. No. 5,454,039; U.S. Pat. No.5,675,652; U.S. Pat. No. 5,835,597; Rogaway, P. and Coppersmith, D., "A Software-Optimized Encryption Algorithm", Proceedings of the 1994 Fast Software Encryption Workshop, Lecture Notes In Computer Science, Volume 809, Springer-Verlag, 1994, pp. 56-63; Rogaway, P. and Coppersmith, D., "A Software-Optimized Encryption Algorithm", Journal of Cryptology, Volume 11, Number 4, Springer-Verlag, 1998, Pages 273-287, and at http://www.cs.ucdavis.edu/~rogaway/papers/seal-abstract.html. Both SEAL and RC4 are discussed in Schneier.

Further, theoretically, a stream cipher is inherently immune to a chosen plaintext attack, and can contain more state information than a block cipher. A block cipher needs to have both encryption and decryption to be secure, and needs to have the avalanche property from the middle to both ends. For example, changing a single bit in the middle of the cipher should change each bit of the input and the output with probability of about 1/2. Also, the stream cipher has the advantage that its outputs are ordered, while a block cipher must be able to efficiently compute every possible output in any possible order. As a result, for many applications stream ciphers are now clearly preferable over block ciphers.

Unfortunately, many stream ciphers have a significant limitation; most cannot efficiently seek to an arbitrary location in their keystream. In this context, seeking to an arbitrary location in the keystream means generating a segment of keystream that is conceptually located an arbitrary number of bits ahead of that portion of keystream that would be generated by ordinary operation of the keystream in its then-current state. This capability is required for numerous practical applications. For example, in a communications protocol that uses unreliable transport, there is no guarantee that data packets of a particular flow will arrive in order, or arrive at all. Examples of such protocols include Internet Protocol (IP), UDP, and RTP. Such protocols commonly experience loss and reorder of packets in practice. Therefore, for a flow that includes successive packets *a, b,* and *c,* a cipher may need to encrypt packet *c* before it encrypts packet *b*. A stream cipher can be used to provide privacy for data communicated using such protocols, if the cipher can seek to the proper location in the keystream for packet *c* based on a sequence number.

Similarly, an encrypted disk partition or file system can use a stream cipher if the cipher supports the seek operation.

These examples do not require the random access capability of a block cipher, in which all inputs are equally simple to compute. Rather, the example applications require the capability to seek into the keystream, with a seek time that is not significant relative to the time required to generate the keystream itself. In this context, “seek” is used in the same sense as used in the POSIX and ANSI C functions for repositioning the offset of a file descriptor.

In one past approach to providing a stream cipher with a seek capability, the state update function is made linear in some field. In this approach, a seek is a composition of linear operations, and therefore is itself linear. This approach is similar to using a block cipher in counter mode, which imposes requirements on the output function that are similar to the requirements on block ciphers.

In an alternative approach, as taken by Rogaway et al. in the design of the SEAL cipher, a special seek function is defined that pseudo-randomly maps an index and a fixed key to an internal state of a keystream generator. Based on this state information, the keystream generator can generate a length of keystream. The keystream for the cipher is defined to be the concatenation of the keystreams generated for each index, with indices in ascending order. Effectively, this approach creates a stream cipher that can seek to some regularly spaced locations in its keystream.

While this approach is satisfactory for many applications, some applications may require the ability to seek to an arbitrary location in the keystream. For example, an encrypted database containing many small records could have this requirement. In addition, the seek function approach adds security requirements. The seek function itself must be secure, and the seek and advance functions must be such that they do not interact in an insecure way.

Based on the foregoing, there is a clear need for an additive stream cipher method that can seek to an arbitrary location in its keystream.

There is a specific need for a stream cipher that provides a keystream seek capability without using a linear state update function, and without a special seek function.

There is also a need to provide such a stream cipher in an embodiment that achieves excellent performance when executed in software implemented for general-purpose computer processors.

SUMMARY OF THE INVENTION

The foregoing needs, and other needs and objects that will become apparent for the following description, are achieved in the present invention, which comprises, in one aspect, a method of automatically generating a keystream segment of an arbitrary location of a complete keystream of an additive stream cipher. A location value that identifies a location of the keystream segment within the complete keystream is received. A state value for a leaf node of a balanced binary tree is created and stored, wherein the leaves of the tree represent the complete keystream and the leaf node represents the keystream segment at the location, by a traversal of the tree from root node to the leaf node wherein a leftward tree branch transition comprises computing a first non-linear function and a rightward tree branch transition comprises computing a second non-linear function. The keystream segment is created and stored by applying a third function to the state value of the leaf node. The entire keystream can be efficiently computed, in order, via a preorder traversal of the tree.

According to another aspect, the invention provides a stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream. A segment of plaintext is received, along with information indicating where the segment of plaintext is located within a complete plaintext. Based on the location information, a keystream segment for that location is created and stored. The segment of plaintext is then enciphered by combining the keystream segment with the plaintext segment using an exclusive OR operation, resulting in creating and storing a segment of ciphertext.

In other aspects, the invention encompasses a computer apparatus, a computer readable medium, and a carrier wave configured to carry out the foregoing steps.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. 1A is a simplified block diagram of a stream cipher.

FIG. 1B is a block diagram of a portion of a balanced binary tree that may be used in one embodiment.

FIG. 1C is a flow diagram of a method of generating a segment of a keystream.

FIG. 1D is a flow diagram of a method of encrypting network communications.

FIG. 2 is a computational graph diagram that illustrates one embodiment of a function $c \circ f$.

FIG. 3 is a computational graph diagram that illustrates one embodiment of a function $c \circ g$.

FIG. 4 is a computational graph diagram that illustrates a concatenation of FIG. 2 and FIG. 3.

FIG. 7 is a block diagram that illustrates a computer system upon which an embodiment may be implemented.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream, and a method of generating an arbitrary segment of keystream, are described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

COMPUTATION TREE APPROACH

An improved additive stream cipher is defined in part by a keystream generating approach that uses a balanced binary tree.

FIG. 1B is a block diagram of a portion of a balanced binary tree that may be used in one embodiment. In the example tree 100 of FIG. 1, root node 102 has fixed state information. Each intermediate node 104 provides a transition path from root node 102 to two of a plurality of leaf nodes 106. Each leaf node 106 represents and is associated with a small length of keystream.

Computation of a keystream segment at a specified location within the complete keystream is carried out by computing from the root node 102 to one of the leaf nodes 106 that is associated with the specified location, then computing the rest of the nodes using a pre-order tree traversal. Thus seeking to a location in the keystream is equivalent to computing the state for a particular leaf 106 in the tree 102, a computation that requires computational effort that is logarithmic in the size of the keystream.

The root node 104 and each leaf node 106 contain *m* bits of state information. A left branch of tree 102 is computed using the nonlinear function

a: GF(2^m) -> GF(2^m)

and a right branch is computed using the nonlinear function

b: GF(2^m) -> GF(2^m)

The outputs of the cipher are computed by applying the function

c: GF(2^m) -> GF(2^r)

to the state information of the leaf nodes, which maps the *n* internal bits to an output of size *r*. Accordingly, in FIG. 1B, leftward transitions from node to node have reference label *a* to indicate that such transitions involve computing function *a*. Similarly, rightward transitions from node to node have reference label *b* to indicate that such transitions involve computing function *b*. Exit paths from leaf nodes 106 have reference label *c* to indicate that such paths involve computing function *c*.

Generating a complete keystream in order is done by a preorder traversal of the entire tree, starting at the root. A binary tree with height *h* contains 2^h - 1 nodes, 2^{h-1} leaves, and 2^{h-1} - 2 edges. In this context, a tree with one node is defined as having height one. Each edge is computed exactly once in a traversal of the tree. Therefore, the present approach can generate keystream at a rate of approximately:

$$\frac{r2^{h-1}}{R_c2^{h-1}+R_{ab}(2^h-2)} \cong \frac{r}{R_c+2R_{ab}}$$

where R_c is the rate at which the function *c* can be evaluated, and R_{ab} is the average rate at which the functions *a* and *b* can be evaluated.

In one embodiment, the tree is implemented by creating and storing a stack in memory of *h* elements, the i^{th} element of which contains the state information of the node on the i^{th} path from the root to the leaf corresponding to the current output. Thus, the computation tree defines a cipher with *mh* bits of internal state, wherein the state update function modifies only a portion of the current state. An important implementation consideration is that the preorder traversal is readily adaptable to parallel processing.

The functions *a* and *b* are comparable to the round functions of a substitution-permutation network. They are the main source of non-linearity in the cipher disclosed herein. The function *c* can be used to hide the internal state, as it has fewer outputs than inputs.

CIPHER PROCESSING FUNCTIONS AND KEY SETUP APPROACH

In one embodiment, a stream cipher that can seek to an arbitrary location in its keystream involved computing functions *a, b,* and *c*, and a key setup approach.

-- IMPROVED CIPHER PROCESSING APPROACH

Some general-purpose central processing units, such as the Intel x86 family of processors, have no more than seven (7) available registers. In one specific embodiment, a cipher approach is optimized to operate with seven registers. Accordingly, in this embodiment, the present approach achieves high throughput on general-purpose processors.

FIG. 1C is a flow diagram of a method of generating a segment of a keystream. In block 120, a location value that identifies a location of a keystream segment within a complete keystream is received. In data communication applications, the location value may comprise a packet identifier that indicates the relative order, within a flow of packets, of a packet that has been received out of order with respect to other packets in the flow.

In block 122, a state value is created and stored by conceptually traversing a balanced binary tree that represents the complete keystream. As stated in block 124, a leaf node in the tree represents the keystream segment. Traversal proceeds from the root node to the leaf node associated with the specified keystream segment, as shown in block 126. To traverse a left branch, a first non-linear function is computed, as shown by block 128. To traverse a right branch, a second non-linear function is computed, as shown by block 130.

In block 132, a third function is applied to the state value, to result in creating and storing the keystream segment.

FIG. 1D is a flow diagram of a method of encrypting network communications.

In block 140, an unencrypted packet is received as part of a stream of network communications. Block 140 may involve receiving one or more packets at any suitable network device or end station, such as a router, switch, gateway, server, etc. Also, the unencrypted information may comprise a portion of a packet, e.g., a payload of a packet wherein the header is treated separately.

In block 142, a location of the packet within a network flow is determined. Block 142 may involve, for example, determining the order of a particular packet within a larger flow of multiple packets, where the packets are received out of order.

In block 144, a keystream segment corresponding to the packet location is created and stored. Block 144 may involve carrying out the steps of FIG. 1C in order to generate a segment of keystream appropriate for the received packet.

In block 146, the packet is encrypted or enciphered using the keystream segment as applied to a stream cipher function. Block 146 may involve encrypting only the payload of a packet.

The process of FIG. 1D and the other processes described herein are applicable to encryption of packets that form portions of voice over Internet Protocol flows in network communications as part of the IPSec protocol, and for the encryption of unreliable network traffic in general, for example, traffic transported using the UDP protocol.

In one embodiment, the number *m* of bits of state per node, and the key size, are specified by parameter values. The parameterizable width enables reduced size versions of the cipher to be studied, enabling numerical experiments on the cipher to be more meaningful. The parameterizable key size is useful to adapt the cipher to different security policies, and to protect against typical plaintext attacks by enabling keys to be increased in size to compensate for the possibility of cryptographic attacks that use precomputation. See, e.g., D. McGrew et al., "Attacks on Linearly Redundant Plaintext and Implications on Internet Security," the Proceedings of the Seventh Annual Workshop on Selected Areas in Cryptography.

Accordingly, in one embodiment, m is restricted to being a multiple of twelve, because of the internal structure of the functions *a* and *b*. In the description herein, *x, y* and *z* denote *n* bit quantities, and each node 104, 106 of tree 100 has $m=3n$ bits of state denoted as $z|y|x$, where the symbol | denotes concatenation, and *n* is a multiple of four

In one embodiment, the state contained by each node 102, 104, 106 of tree 100 contains a unique node number to ensure that two distinct nodes cannot have the same state. The node number is denoted as *z*, and counted as part of the state of the node to which it corresponds. The node number of the root node 102 is defined as one. If the node number of a node is *z*, that of its left child is *2z* and that of its right child is *2z+1*. Functions of the cipher use substitution, rotation, and addition modulo 2^n. The "filter" function *c* is defined as the linear reduction of 2*n* bits to 2 bits. The branching functions *a* and *b* are defined as the composition of a diffusion function *d* with the nonlinear "confusion" functions *f* and *g*. Thus,

$$a = f \circ d$$

and

$$b = g \circ d$$

The definitions of functions *f, g, d,* and *c* are

$$f(z\,|\,y\,|\,x) \;= 2z\,|\,S\,(R\,(S\,(R\,(y)\,)\,)\,)\,|\,L\,(S\,(L\,(S\,(x)\,)\,)\,)$$

$$g(z\,|\,y\,|\,x) \;= 2z{+}1|\,L\,(S\,(L\,(S\,(y)\,)\,)\,)\,|\,S\,(R\,(S\,(R\,(\Box x)\,)\,)\,)$$

$$d(z\,|\,y\,|\,x) \;= z \;\;|\,x + y + z \;\;|\,2x + y + z$$

$$c(z\,|\,y\,|\,x) = x \oplus y$$

where integer addition modulo two to the power n is denoted as +, bitwise exclusive-or is denoted as $\oplus$, and bitwise complementation is denoted as □. The functions *R* and *L* indicate rotation by $n/4$ bits and to the right and left, respectively (where "right" means in the direction of the least significant bit). The nonlinear function S is implemented as a key-dependent substitution table.

The filter function c is a bitwise exclusive or of x and y. The composition of c with f and g is shown in FIG. 2, FIG. 3. Specifically, FIG. 2 is a computational graph diagram that illustrates one embodiment of the function $c \circ f$, as indicated by reference numeral 200, and FIG. 3 is a computational graph diagram that illustrates one embodiment of the function $c \circ g$, designated by reference numeral 300. FIG. 4 is a computational graph diagram that illustrates a concatenation of FIG. 2 and FIG. 3 (reference numeral 400). FIG. 5 is a computational graph diagram that illustrates one embodiment of the function $f \circ d$ (reference numeral 500). FIG. 6 is a computational graph diagram that illustrates one embodiment of the function $g \circ d$, as indicated by reference numeral 600.

In FIG. 2, FIG. 3, FIG. 4, FIG. 5, and FIG. 6, inputs $X0, X1, X2, X3$ represent the bytes of x, where $X0$ is the least significant byte, $X1$ is the second least significant, and so on. Similarly, inputs $Y0$, $Y1$, $Y2$, $Y3$ represent the bytes of variable y, from least significant to next least significant, and inputs $Z0$, $Z1$, $Z2$, $Z3$ represent bytes of variable z. The function S depends only on the lowest $n/4$ bits of its argument, and can be viewed as four component functions $S0$, $S1$, $S2$, $S3$, as indicated by like labeled blocks in FIG. 2, FIG. 3, FIG. 4, FIG. 5, and FIG. 6. A key setup approach is defined such that component functions $S0$, $S1$, $S2$, $S3$ are invertible.

FIG. 2 and FIG. 3 illustrate how the output of the cipher disclosed herein is derived from the state of the leaf nodes using one embodiment of f, g, and c. The structures of f, g, and c are optimized, in this embodiment, to resist up and down attacks. Referring to FIG. 3, the variables at the top and bottom (z, w) represent two adjacent n-bit outputs of the cipher. The state at the middle of the network, represented by x, y, is the output of the function d for each of the leaves.

The manner in which the functions *f* and *g* use a substitution table is similar to that of the block cipher WAKE, which is described in B. Schneier, supra, at 400,. However, in the approach described herein, greater parallelism and lower register usage is achieved by using the substitution table to modify the value of its argument.

-- KEY SETUP

Key setup consists of generating four invertible functions, and storing them in concatenated form in a substitution table. The substitution operation herein maps a single variable x to $x \oplus s(x)$, for some substitution table s that depends only on the lowest 8 bits of x. Thus, invertibility requires that the function mapping the lowest 8 bits of x to lowest 8 bits of $x \oplus s(x)$ be invertible. This is achieved by pseudorandomly selecting an invertible function mapping 8 bits to 8 bits, then composing that function with the bitwise exclusive-or function. Invertible functions are pseudorandomly generated by swapping elements at random in the table, an approach similar to that of the RC4 next-state function.

Detailed pseudocode for this method follows, in which S0, S1, S2, and S3 are the substitution tables described above, 'key' represents the raw key, and i, j, k, and 'index' are integers between zero and 255. The symbol % denotes the modulus operator. The number of bytes in a key is represented here by the symbol bytes_in_key, which is a parameter of this method.

PSEUDOCODE FOR KEY SETUP

```
1. (initialize substitution tables to zero)
for i from 0 to 255
        set S0[i] to 0
        set S1[i] to 0
        set S2[i] to 0
        set S3[i] to 0
endfor
```

2. (set substitution tables to key-dependent permutations)

```
for j from 0 to 255
          for i from 0 to 255
          set S3[i] to S2[i]
          set S2[i] to S1[i]
          set S1[i] to S0[i]
          set S0[i] to i
          endfor
          set index to j
          for k from 1 to num_passes
          for i from 0 to 255
          set index to index + (key[i % bytes_in_key] + S0[i]) % 255
          set tmp to S0[i]
          set S0[i] to S0[index]
          set S0[index] to tmp
          set tmp to S1[i]
          set S1[i] to S1[index]
          set S1[index] to tmp
          set tmp to S2[i]
          set S2[i] to S2[index]
          set S2[index] to tmp
          set tmp to S3[i]
          set S3[i] to S3[index]
          set S3[index] to tmp
          endfor
          endfor
endfor
```

3. (set S0 equal to the xor of itself with the identity permutation)

```
for i from 0 to 255
          set S0[i] to the value of S0[i] exclusive-ored with i
```

```
endfor
```

C language computer program source code for an example of a key setup approach is presented in Appendix A.

-- PERFORMANCE

The functions *f* and *g* can each be computed in seven instructions on a general purpose CPU that can issue two instructions simultaneously, using five registers (holding *x*, *y*, *z*, a temporary variable, and a pointer to the substitution table). The theoretical encryption rate is about fifteen cycles per byte (using two execution units).

HARDWARE OVERVIEW

FIG. 7 is a block diagram that illustrates a computer system 700 upon which an embodiment of the invention may be implemented.

Computer system 700 includes a bus 702 or other communication mechanism for communicating information, and a processor 704 coupled with bus 702 for processing information. Computer system 700 also includes a main memory 706, such as a random access memory ("RAM") or other dynamic storage device, coupled to bus 702 for storing information and instructions to be executed by processor 704. Main memory 706 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 704. Computer system 700 further includes a read only memory ("ROM") 708 or other static storage device coupled to bus 702 for storing static information and instructions for processor 704. A storage device 710, such as a magnetic disk or optical disk, is provided and coupled to bus 702 for storing information and instructions.

Computer system 700 may be coupled via bus 702 to a display 712, such as a cathode ray tube ("CRT"), for displaying information to a computer user. An input device 714, including alphanumeric and other keys, is coupled to bus 702 for communicating information and command selections to processor 704. Another type of user input device is cursor control 716, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 704 and for controlling cursor movement on display 712. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

The invention is related to the use of computer system 700 for performing a stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream, and a method of generating an arbitrary segment of keystream. According to one embodiment of the invention, performing a stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream, and a method of generating an arbitrary segment of keystream, are provided by computer system 700 in response to processor 704 executing one or more sequences of one or more instructions contained in main memory 706. Such instructions may be read into main memory 706 from another computer-readable medium, such as storage device 710. Execution of the sequences of instructions contained in main memory 706 causes processor 704 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 704 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 710. Volatile media includes dynamic memory, such as main memory 706. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 702. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 704 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 700 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 702. Bus 702 carries the data to main memory 706, from which processor 704 retrieves and executes the instructions. The instructions received by main memory 706 may optionally be stored on storage device 710 either before or after execution by processor 704.

Computer system 700 also includes a communication interface 718 coupled to bus 702. Communication interface 718 provides a two-way data communication coupling to a network link 720 that is connected to a local network 722. For example, communication interface 718 may be an integrated services digital network ("ISDN") card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 718 may be a local area network ("LAN") card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 718 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 720 typically provides data communication through one or more networks to other data devices. For example, network link 720 may provide a connection through local network 722 to a host computer 724 or to data equipment operated by an Internet Service Provider ("ISP") 726. ISP 726 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 728. Local network 722 and Internet 728 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 720 and through communication interface 718, which carry the digital data to and from computer system 700, are exemplary forms of carrier waves transporting the information.

Computer system 700 can send messages and receive data, including program code, through the network(s), network link 720 and communication interface 718. In the Internet example, a server 730 might transmit a requested code for an application program through Internet 728, ISP 726, local network 722 and communication interface 718. In accordance with the invention, one such downloaded application provides for carrying out a stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream, and a method of generating an arbitrary segment of keystream.

The received code may be executed by processor 704 as it is received, and/or stored in storage device 710, or other non-volatile storage for later execution. In this manner, computer system 700 may obtain application code in the form of a carrier wave.

OTHER EMBODIMENTS AND MODIFICATIONS

Accordingly, a fast stream cipher that can efficiently seek to arbitrary locations in its keystream has been described. Unlike RC4, which cannot seek at all, and SEAL, which can seek only to a limited number of locations in the keystream, the approach described herein can seek to any location in the keystream. The cipher can efficiently seek to arbitrary locations in its keystream despite the fact that it does not use a linear state update function. This property is apparent from the architecture of the cipher in which its computation graph may be represented by a tree. Keystream can be efficiently generated using a preorder traversal of the tree, and the tree structure lends itself to parallelization.

Implementations may be embodied in one or more hardware circuits, one or more field programmable gate arrays (FPGAs), or in one or more software programs or processes. Any such implementation may be parallelized, offering significant gains in performance, as a result of the tree structure of the computations.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.